Safe Ways in Models for Safety Critical Systems
2004 IEEE International Conference on Intelligent Engineering Systems

Miklós Szíjártó
Department of Computer Science
Széchenyi István University
Egyetem tér 1, H-9026 Győr
Hungary
szijarto@sze.hu

Gábor Kallós
Department of Computer Science
Széchenyi István University
Egyetem tér 1, H-9026 Győr
Hungary
kallos@sze.hu

Tamás Hajba
Department of Mathematics
Széchenyi István University
Egyetem tér 1, H-9026 Győr
Hungary
hajbat@sze.hu

Abstract – **In this paper, after summarizing our former results we introduce a theoretical model for safety-critical systems, in which the danger-degree of the points can be measured. We present an algorithm to determine the safest way between two given points in polynomial time. We illustrate the theoretical discussion with some examples and applications.**

I. INTRODUCTION

A. Safety-Critical Systems

The engineering systems can be classified in three main categories:

1) ordinary: No special requirements.

2) reliable: Enhanced reliability in system's services and operation.

3) safety-critical: The abnormal operation might endanger human lives and cause significant material damage.

The development of safety-critical systems is a complex, interdisciplinal field, which includes parts of

1) mathematics;

2) software engineering and concurrent programming;

3) mechanical and electronical engineering.

These fields are applied in different parts of the development's lifecycle, which includes – according to the classical theory – requirement definition and analysis, specification, design, construction, verification and validation, maintance.

1) We can determine the safety of the system from mathematical point-of-view, using theoretical models; or apply mathematically based techniques (e.g. formal methods) for system specification and development [4]. A general problem with these results is that they are mostly academic. Applications are known only in specialized development centers. Some problems, which these methods have to overcome: real-life systems are too large, handling and modeling them is very complicated, they often have non-deterministic character.

2) The system usually has software parts, so applying methods of software engineering are unavoidable. Handling of safety-critical systems needs special approaches; we have to guarantee reliability, safety and some self-protection. During the lifecycle – compared with other systems – there are some stressed fields, as risk-management in specification, fail-safety, -tolerance and -avoidance in design, moreover verification and validation is more specific [5]. Usually, in design and construction, we have to apply methods of concurrent programming, details and some general problems can be found e.g. in [2].

3) From an engineer's point-of-view, safety is a technical and managerial issue. A general principle is, that we cannot ensure safety and quality after the construction (e.g. with tests), it must be built in. So, the whole lifecycle must be carefully controlled [4]. In software engineering, there is an American standard to measure the built in quality, the so-called CMM model [5].

Especially in railway traffic, many subsystems are safety-critical. A general investigation about this topic with theoretical and practical questions can be found in [1].

B. Mathematical models

Of course, even with the best planning and production methods, absolute safety is an unobtainable goal, at least for a wide range of systems, if not for all. Besides the hidden failures, we have to consider e.g. human errors and environment changes.

A possible way of handling this situation is to "predict" the closeness of dangerous system states, and avoid them, if possible. There are several references in this topic but generally well applicable results are yet unknown [4].

II. FORMER RESULTS

In our former papers [6] and [7] we presented a mathematical model to compute the "closeness" of critical (dangerous) conditions in safety-critical systems, using graphs and Petri nets. The theory of the distance and probability model is a hopeful result, although we have examined the possibility of their practical applications only in some simple cases.

Let us introduce a graph in the following manner. We denote the status (or condition) of the system in a given moment with a node. Changes are represented by directed edges (transitions) to other nodes. Cycles are allowed, too.

A. Distance model

In this model we initiated distances for

edges: the distances are non-negative numbers;

ways: going on consecutive edges the distances are summarized.

Between two nodes, we have to consider all ways connecting these two nodes. The resultant distance must be (following real life considerations) less than or equal to the minimum of the distances. Possible choices are simply the minimum or the harmonic average.

Example: Let us consider a simple system, where from condition c we can reach condition a in two ways, having

distances d_1 and d_2, respectively. Thus, the resultant distance can be

$$d = \min(d_1, d_2), \text{ or} \tag{1}$$

$$d = \frac{1}{\frac{1}{d_1} + \frac{1}{d_2}} = \frac{d_1 \cdot d_2}{d_1 + d_2} \tag{2}$$

B. Probability model

In this model, we label the edges of the graph with probabilities. The probability for an edge from node i to j is simply the probability p of the transition from i to j with $0 < p < 1$. For ways, going on consecutive edges the probabilities are multiplied.

Between two nodes, the resultant probability is the sum of the probabilities of all ways connecting these two nodes, with the criterion that it must be less than or equal to one.

Example: In the simple system investigated above, the resultant probability p from c to a is

$$p = \prod_i p_i + \prod_j p_j \tag{3}$$

where the products are calculated for the first and the second way with indices i and j, respectively.

C. Additional results

Sometimes the distances, in other cases the probabilities are given, that is why we attempted to connect the two models. We can change from probability to distance model with some kind of logarithmic function, while the reverse direction with some kind of exponential function, but several problems are yet unsolved.

Finally, we mentioned that an algorithm can be applied in our models to determine the distances correctly.

III. GOALS AND PRELIMINARIES

Our goal in this paper is to present a method – in a slightly modified model – to determine the "safest" way between two nodes (this way is not necessarily unique). After it, some investigations will be made to analyze the results.

Let $G = (V, E)$ be a directed graph (which is considered to model a safety-critical system), where V is the set of nodes, E is the set of edges (transitions). Let D be the set of death-points, i.e. the points, which we want to avoid to guarantee safe operation. Let us give two points t_1 and t_2, between which we are interested in the safest way.

The safest way for us is a way, which is the farthermost from any of the death-points (see the definition below).

IV. THE ALGORITHM

Our algorithm determines the desired way in the following steps:

1) We specify to every point v a number dang(v), which measures its danger-degree.

2) We define the danger-degree of a way by summarizing the nodes' danger-degree.

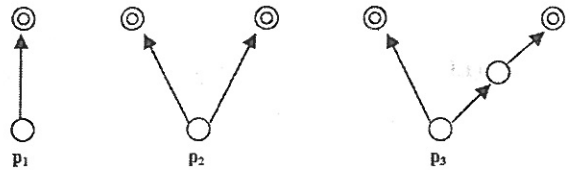

Fig. 1. Examples to illustrate the introduction of danger-degree

3) We determine the right way by using Dijsktra's method.

Let us denote the length of the minimum-length way from node v to w by $\delta(v, w)$.

Definition:

Let $D = (d_1, d_2, \ldots, d_n)$ be the set of death-points. Let the danger-degree of the death-points be infinite. For an arbitrary point $v \in V - D$ let

$$dang(v) = \sum_{d_i \in D} \frac{1}{\delta(v, d_i)} \tag{4}$$

The motivation behind this definition is as follows.

If a point has only one neighbor, which is a death-point, then its danger-degree is 1 (unit) ("Fig. 1", first example).

Following real life considerations, we have to ensure that if there are several death-point-neighbors, then the danger-degree is larger. E.g. for the second example in "Fig. 1", the danger-degree is 2. If the death-point is far away, the danger-degree is smaller. Applying the definition for the third example in "Fig. 1", we can see that dang(p_3) = 1.5.

Step1 (general determination of the danger-degree)

Let us reverse the edges of the graph. From death-points $d_1, d_2, \ldots, d_n$ let us apply Dijkstra's algorithm (finding the minimal way, see e.g. in [3]) for every d_i, to determine values $\delta(d_i, v)$, for every $v \in V - D$.

Remarks

a) Since the algorithm works for graphs having edge-weights, before the procedure we label the edges with weights 1.

b) In the graphs – for correct execution of the algorithm – cycles are allowed, but negative weights are not, our graphs are clearly such types.

c) We could work in step1 in "normal" (not reverse) manner, too; namely we could start the algorithm from every non-death-point. However, with this choice finally we would get the distances between all of the pairs of points, which are clearly not necessary.

d) Its worthwhile considering the efficiency of step1. Taking into account the considerations given in [3], the execution needs at most cnE steps, where n is the number of death-points, E is the number of edges and c is a constant.

Step2)

Definition:

For a way $P = (v_1, v_2, \ldots, v_k)$ let

$$dang(P) = \sum_{i=1}^{k} dang(v_i) \tag{5}$$

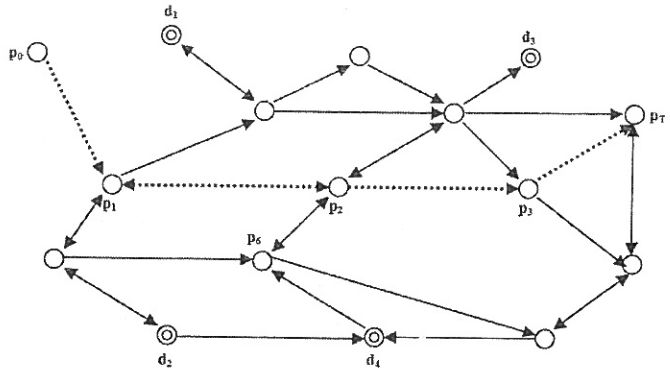

Fig. 2. Sample system for the presentation of the algorithm

From the definition, the smaller the sum dang(P) is, the "better" the way is.

Step3)

To determine the safest way, we will apply Dijkstra's algorithm. To achieve this, from the original graph we build a new one (which is essentially equivalent), which has edge-weights.

Let us make a directed graph $G_2 = (V_2, E_2)$, for which initially $V_2 = V$, $E_2 = E$. Let us moreover label the edges $e = (u, v)$ of G_2 by $c(e)$ = dang(v). For a way P let us introduce $c(P)$ by simply summarizing the $c(e)$-s of the consecutive edges. Now for a way $P = (v_1, v_2, \ldots, v_l)$ we have $c(P)$ = dang(P) - dang(v_1).

This means that if for a way P from node t_1 to t_2 in the original graph G the quantity dang(p) is minimal, then this is a minimum-length way in G_2, regarding the weight-function c, and the reverse direction holds, too.

Thus, if we apply Dijkstra's algorithm in G_2 to determine the minimum-length way from node t_1 to t_2, this will be a way having minimal danger-degree in G.

Remark

Considering the efficiency of step3, we deduce that now the execution needs at most cE_2 steps, where E_2 is the number of edges in G_2 (the same as E) and c is a constant. Taking into account the former result for step1, we can conclude that our method is polynomial.

V. AN EXAMPLE

To demonstrate the execution of the algorithm, let us choose a system, which can be modeled by the directed graph presented in "Fig. 2".

We are interested in the safest way from p_0 to p_t. Applying the first step of the algorithm, using (4) we got the danger-degree of all of the points. For p_0, we have

$$dang(p_0) = \frac{1}{3} + \frac{1}{3} + \frac{1}{4} + \frac{1}{4} = \frac{7}{6} \approx 1.116, \quad (6)$$

where the values in the expression are given for death-points d_1, d_2, d_3 and d_4, respectively. E.g. for point d_1, the minimum-length way is $P = (p_0, p_1, p_5, d_1)$, so its length is 3. Similarly, for the other death-points, the lengths are 3, 4 and 4.

For some other points – similarly as in (6) – we have dang(p_1) = 5/3, dang(p_2) = 5/3, dang(p_3) = 61/84, dang(p_6) = 4/3 and dang(p_t) = 61/84.

The result for dang(p_3) comes from

$$dang(p_3) = \frac{1}{3} + \frac{1}{8} + \frac{1}{8} + \frac{1}{7}. \quad (7)$$

For other points, the values are omitted.

Thus, applying the third step of the algorithm, we will get the safest way as $P_s = (p_0, p_1, p_2, p_3, p_t)$. Illustrating one sub step let us show why point p_3 was chosen after p_2, why not p_6. We know that dang(p_3) is less than dang(p_6) and from p_6 to reach p_t at least two more nodes are needed with some positive danger-degrees.

The way's danger-degree is

$$dang(P_s) = \frac{7}{6} + \frac{5}{3} + \frac{5}{3} + \frac{61}{84} + \frac{61}{84} = \frac{125}{21} \approx 5.952. \quad (8)$$

VI. REAL-LIFE APPLICATIONS

Considering the real-life situation, sometimes we face a problem, that only a set of the possible starting or terminal points are given. In this case, we have to determine the endpoints. We present a possible method based on our algorithm.

Let us denote the set of the possible starting points and the set of the possible terminal points with $S = (s_1, s_2, \ldots, s_n)$ and $T = (t_1, t_2, \ldots, t_m)$, respectively. For every pair (s_i, t_j) we determine the way from s_i to t_j with minimal danger-degree, denoting its value by dang(i, j). Finally, we choose our endpoints s_i and t_j, for which dang(i, j) is minimal.

A worse situation is, when only a set of the starting points is given, and we cannot predict exactly which endpoint we should arrive later (i.e. we have no possibility of choosing among the terminal points).

Using the "pessimistic" approach, for safety we would like to minimize the danger-degree even in the worst case. To do this, for every pair (s_i, t_j) we determine the way from s_i to t_j with minimal danger-degree, denoting its value by dang(i, j). Finally, we choose our endpoints s and t, so that

$$\min_i \max_j dang(i, j) = dang(s, t). \quad (9)$$

If we have more information about the endpoints, e.g. the probability of their choice is given (in the simplest case all of them are equal), then for every starting point we are able to determine the expected value of the danger-degree of the way to the terminal points. We choose the starting point for which this expected value is minimal.

VII. EVALUATION

It can be seen that even in our small example the comparison of the "goodness" of two different ways is not an easy task to do. In real-life applications mentioned above the situation is much more complicated. However, the use of the danger-degree gives an exact method in simple and complicated cases, and sometimes it helps us to find elegant solutions.

VIII. REFERENCES

[1] Benyó B., Héray T., Kallós G., Keresztes P., Pataricza A., Rózsa G., Szíjártó M., Sziray J., "Vasúti

Biztonság-Kritikus Rendszerek Verifikációja és Validációja," *Vezetékek Világa*, 2000/3, pp. 15-19.

[2] A. Burns and G. Davies, *Concurrent Programming*, Addison Wesley Publishers Limited, 1993.

[3] T. Cormen, C. Leierson, R. Rivest, *Introduction to Algorithms*, The Massachusetts Institute of Technology, 1990.

[4] F. Redmill and T. Anderson (Eds.), *Safety-Critical Systems: current issues, techniques and standards*, Chapman & Hall, 1993.

[5] I. Sommerville, *Software Engineering (sixth edition)*, Pearson Education Limited, 2002.

[6] M. Szíjártó, D. Grőger and G. Kallós, "A Qualitative Model for Conditions in Safety-Critical Systems", *Hungarian Electronic Journal of Sciences*, 1999, (http://heja.szif.hu/ANM/).

[7] M. Szíjártó, D. Grőger and G. Kallós, "A Distance Model for Safety-Critical Systems", *Periodica Polytechnica Electrical Engineering*, vol. 45, no. 2, 2001, pp. 109-118.